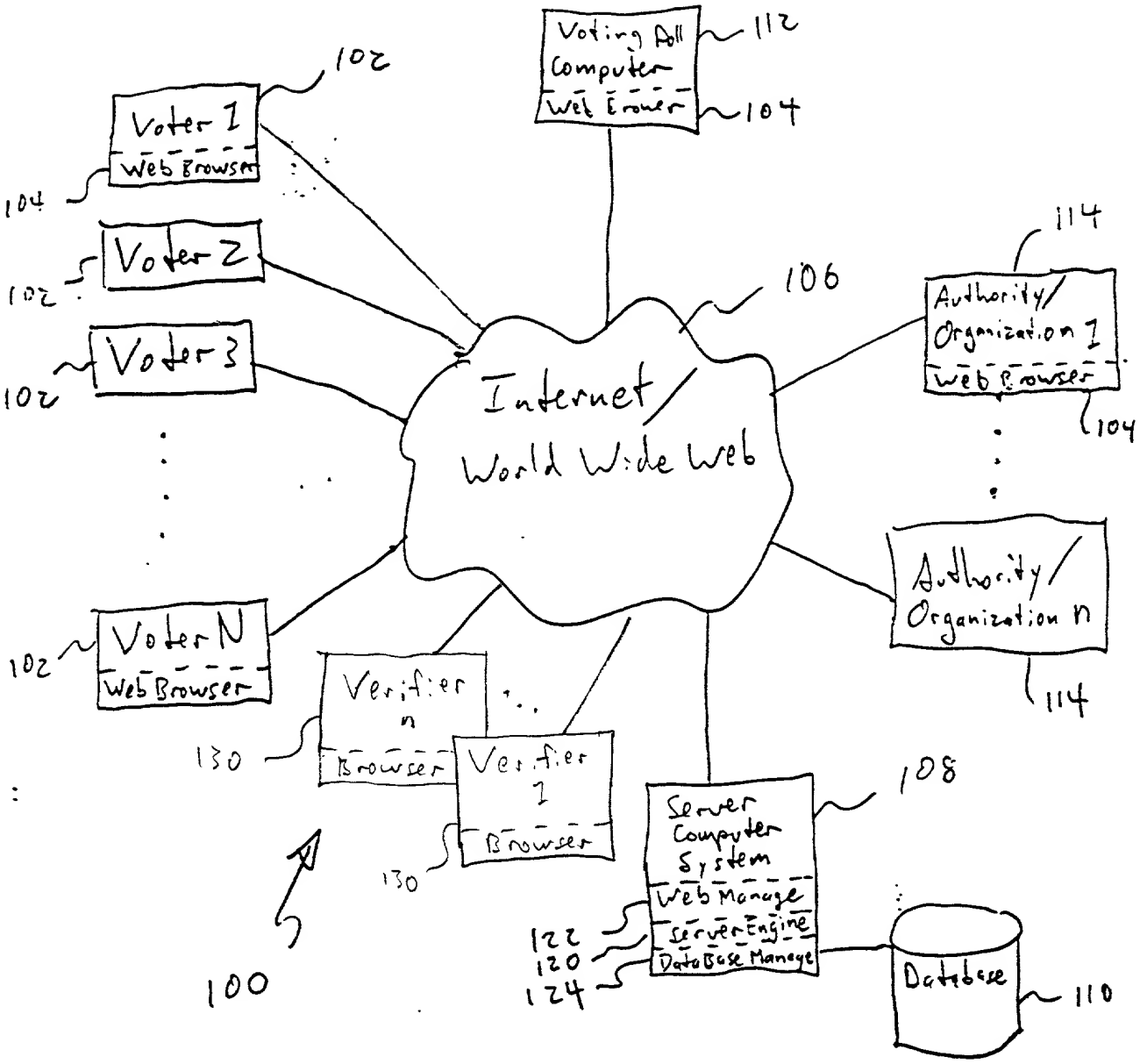

Voting Booth Computer
Web Browser
112
104
102
Voter 1
Web Browser
104
Voter 2
102
Voter 3
102
Voter N
Web Browser
102
106
Internet / World Wide Web
114
Authority / Organization 1
Web Browser
104
Authority / Organization n
114
Verifier n
Browser
130
Verifier 1
Browser
130
100
Server Computer System
Web Manage
Server Engine
Data Base Manage
108
122
120
124
Database
110

FIG. 1

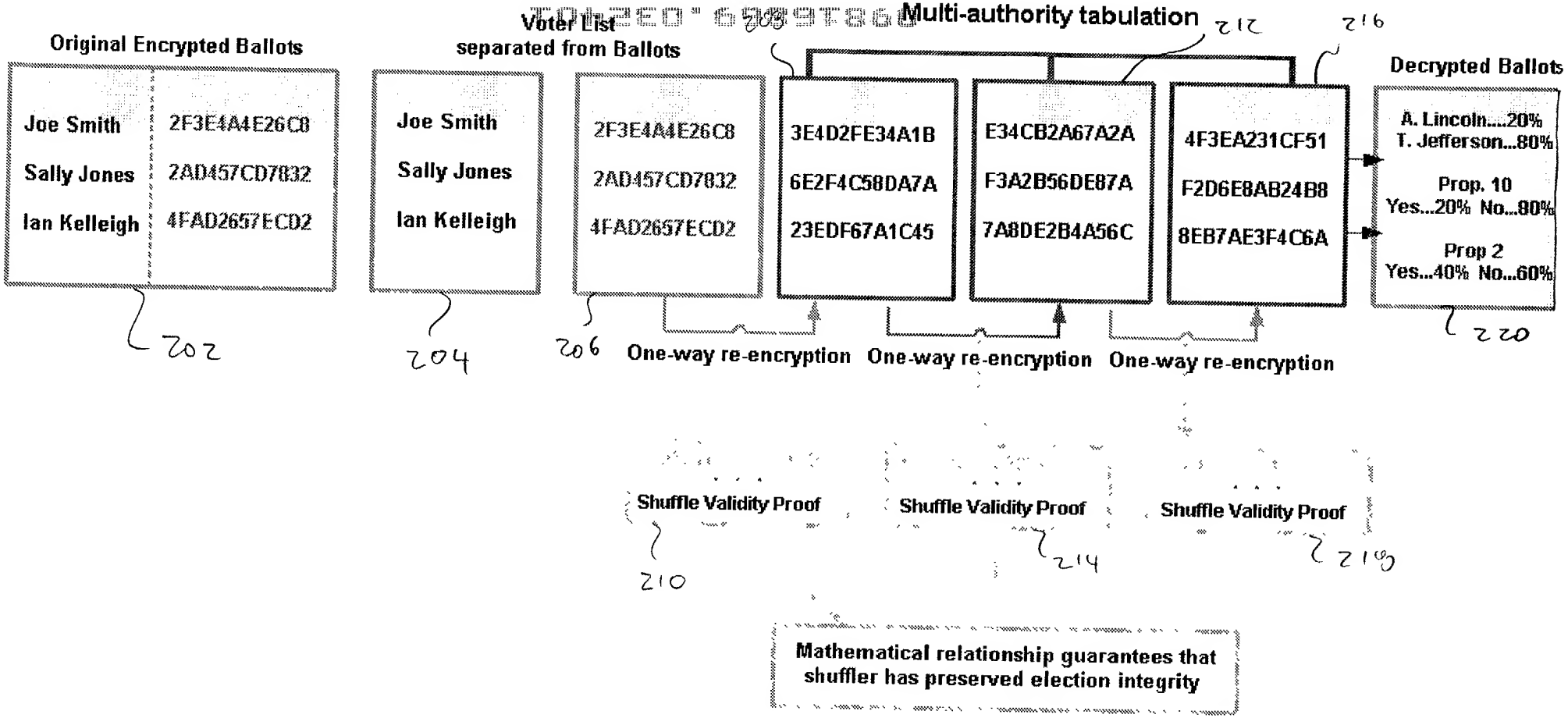
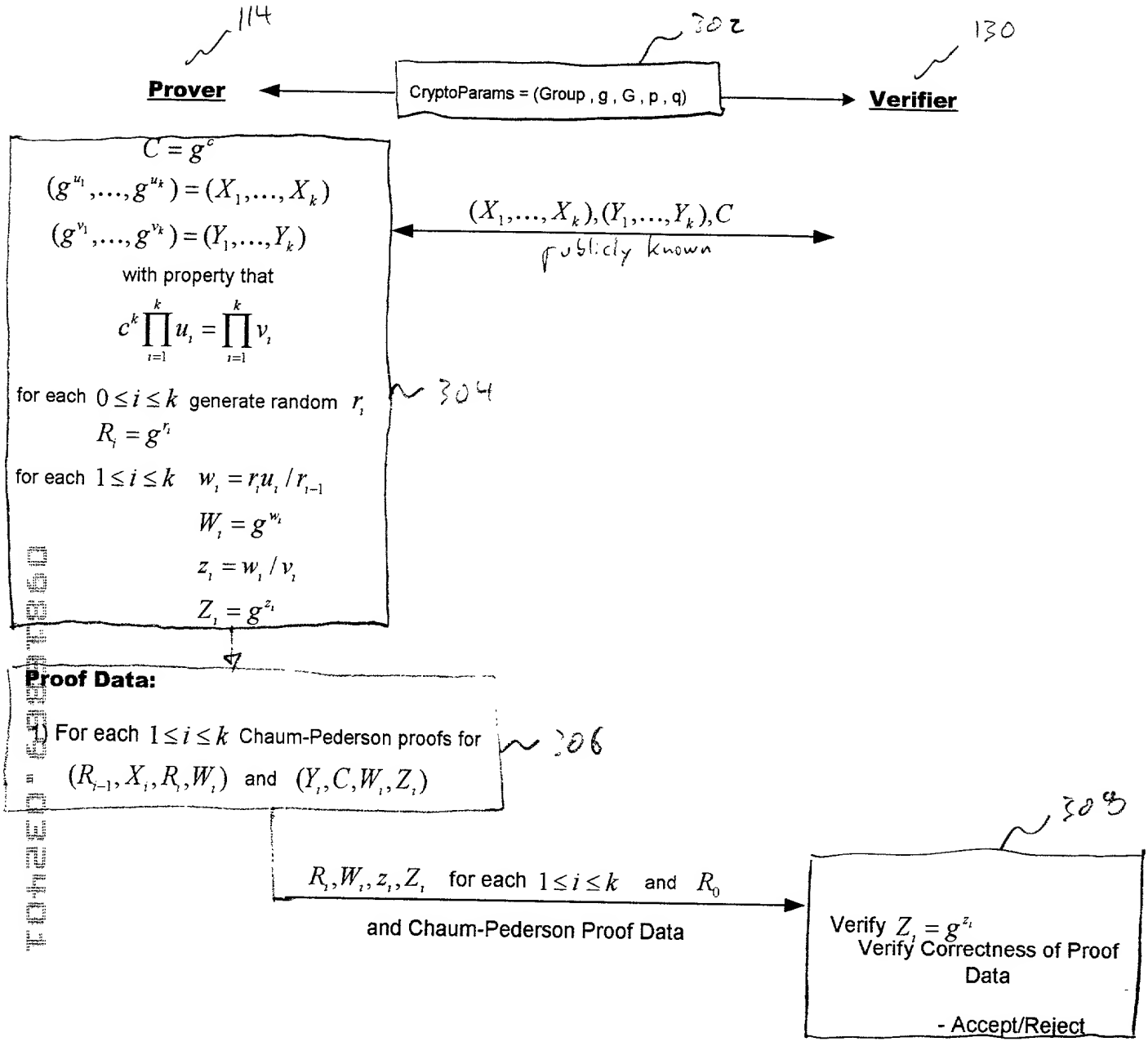

Original Encrypted Ballots
Joe Smith
Sally Jones
Ian Kelleigh
2F3E4A4E26C8
2AD457CD7832
4FAD2657ECD2
202
Voter List
separated from Ballots
Joe Smith
Sally Jones
Ian Kelleigh
204
2F3E4A4E26C8
2AD457CD7832
4FAD2657ECD2
206
Multi-authority tabulation
212
216
3E4D2FE34A1B
6E2F4C58DA7A
23EDF67A1C45
E34CB2A67A2A
F3A2B56DE87A
7A8DE2B4A56C
4F3EA231CF51
F2D6E8AB24B8
8EB7AE3F4C6A
Decrypted Ballots
A. Lincoln....20%
T. Jefferson...80%
Prop. 10
Yes...20% No...80%
Prop 2
Yes...40% No...60%
220
One-way re-encryption
One-way re-encryption
One-way re-encryption
Shuffle Validity Proof
210
Shuffle Validity Proof
214
Shuffle Validity Proof
218
Mathematical relationship guarantees that shuffler has preserved election integrity
200

FIG 2

Scaled Iterated Logarithmic Multiplication Proof



300

FIG 3

Simple Shuffle
(Shuffler knows exponents)

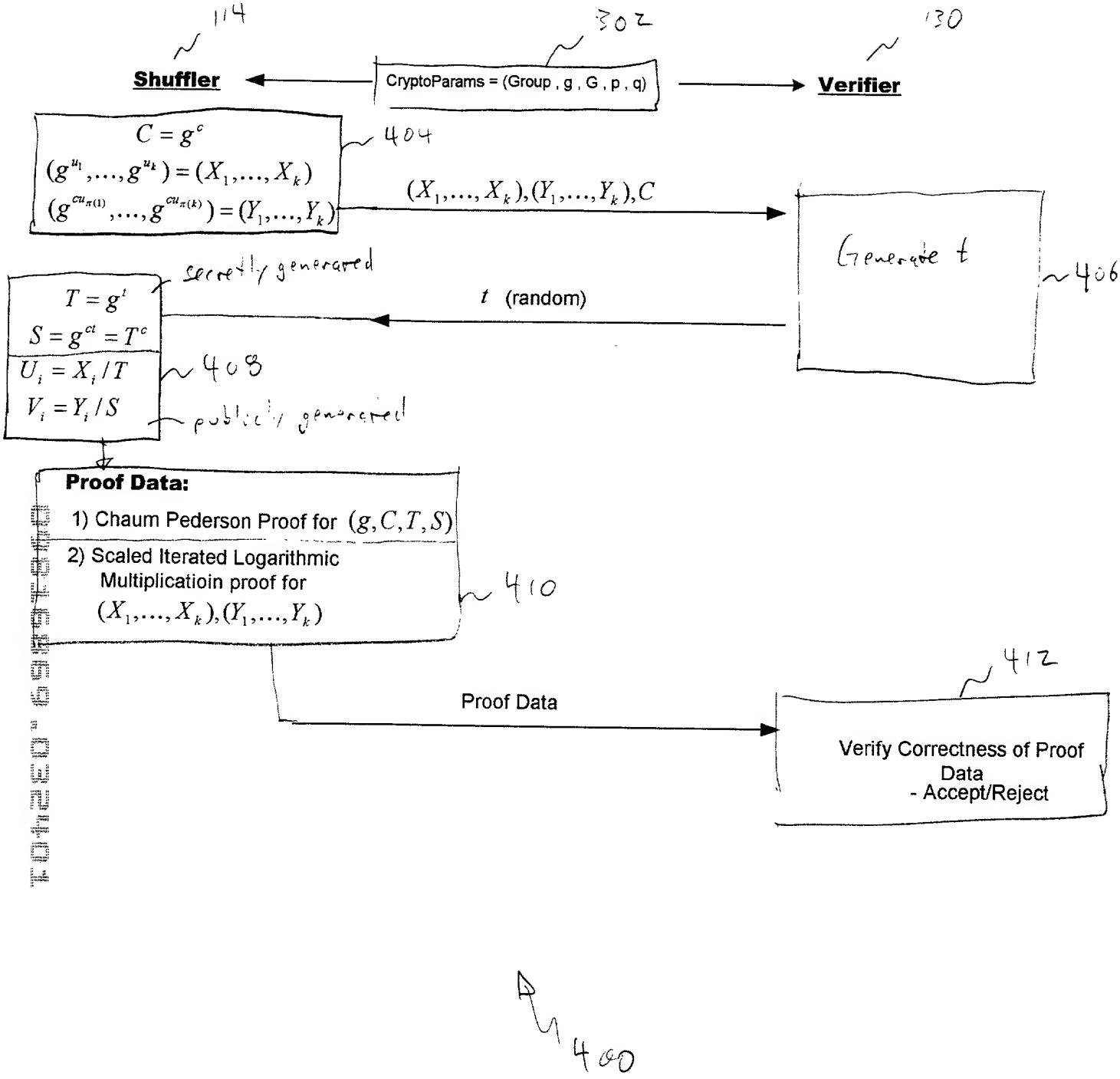


Fig 4

General Shuffle
(Shuffler does not know exponents)

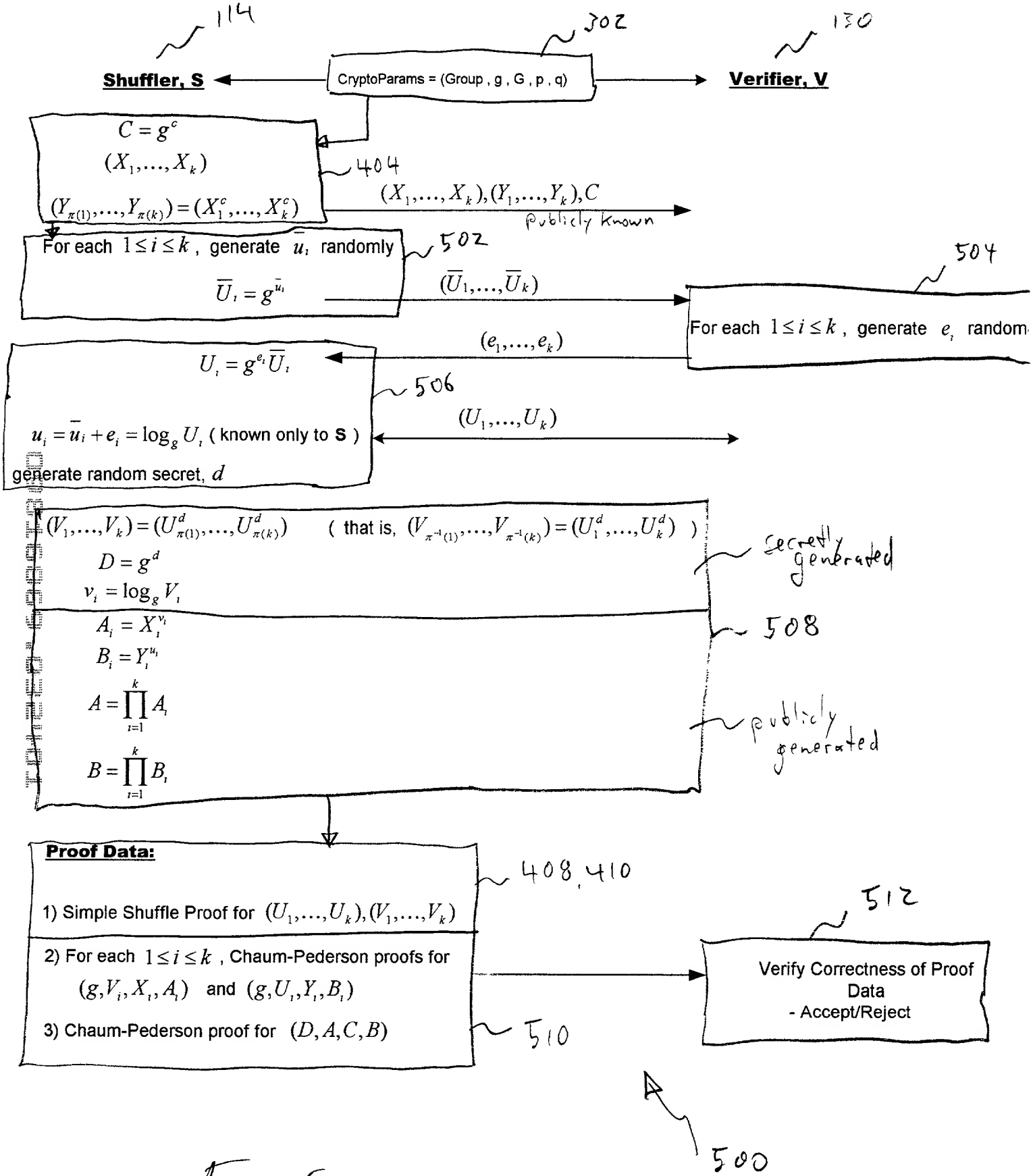


FIG 5

Anonymous Certificate Distribution (Protocol Variant 1)

600

Initialization

302

CryptoParams = (Group , g , p , q) -- published

604

$K = H$ = set of standard public keys $= \{h_1, \ldots, h_k\}$

Registrants each know corresponding private key, s_j such that $h_j = g^{s_j}$ 606

$G = g$ 608

610

Optional Randomization by Authorities

In sequence, each authority performs verifiable shuffle on H using $(G, C = G^e)$ as the shuffle commitment, and returns the shuffled set, H', along with the shuffle verification transcript, $T(H, H', G, C)$.

If the verification transcript is correct, Registration Server performs the substitutions

$$G = C \qquad H = H'$$

and stores the previous values, along with the shuffle verification transcript for audit purposes.

(This can be performed as part of initialization, and/or, at any intermidiate stage of anonymous certificate generation.)

Anonymous Certificate Request and Generation Phase (each registrant in turn)

Registrant		***Registration Server***
612 Generate Request	anonymous authentication request →	
		Retreive G, H 614
	← G, H	
1) compute shuffle (and verification transcript)		
	$T(H, H', G, C)$, $e = cs$ and index, $1 \le j \le k$ →	618
2) Generate PKI Certificate Request with "random identifying information"	R = PKI Certificate Request →	1) Check shuffle verification transcript 2) Check $h'_j = G^e$
3) Safely store corresponding private key for this request. 616		*If both checks pass* 3) Set $K = H = H' - \{h'_j\}$ $G = C$ $k = k - 1$ 4) Store $T(H, H', G, C)$ for audit purposes
	← $\Omega(R)$	5) Digitally sign R thereby creating PKI Certificate, $\Omega(R)$
	← deny request	*Else, if any check fails*

Loop to begining of this phase (ready for next anonymous authentication request)

Fig 6

Anonymous Certificate Distribution
(Protocol Variant 2)

700

Initialization

302

CryptoParams = (Group , g , p , q) -- published

704

$K =$ set of standard public key pairs $= \{(g_1, h_1), \ldots, (g_k, h_k)\}$

$H \subset K \qquad J = K - H$

606

Registrants each know corresponding private key, s_j such that $h_j = g_j^{s_j}$

710

Optional Randomization by Authorities

In sequence, each authority performs verifiable shuffle (of pairs) on K using $(g, C = g^c)$ as the suffle commitment, and returns the shuffled set, K' , along with the shuffle verification transcript, $T(K, K', G, C)$

If the verification transcript is correct, Registration Server performs the substitution

$$K = K'$$

and stores the previous values, along with the shuffle verification transcript for audit purposes.

(This can be performed as part of initialization, and/or, at any intermidiate stage of anonymous certificate distribution.)

Anonymous Certificate Request and Generation Phase (each registrant in turn)

Registrant		***Registration Server***
Generate Request 712	anonymous authentication request →	
		Retrieve H 714
1) Select subset $M \subset H$ of size $k' \leq k$ and set $M' = H - M$	← H (contains registrant's public key)	
2) Compute shuffle, H', of M and verification transcript)		
	$T(M, H', g, C)$, P →	718
	$R =$ PKI Certificate Request	
3) Generate zero knowledge proof, P that registrant knows exponent s such that $(g'_j)^s = h'_j \in H'$ for specified index, $1 \leq j \leq k'$		1) Check shuffle verification transcript 2) Check P
		If both checks pass
		3) Set $K = J \cup M' \cup (H' - \{(g'_j, h'_j)\})$ $k = k - 1$
4) Generate PKI Certificate Request with "random identifying information" 716		4) Store $T(M, H', g, C)$ for audit purposes
5) Safely store private key corresponding to this certificate request		
	← $\Omega(R)$	5) Digitally sign R thereby creating PKI Certificate, $\Omega(R)$
	← deny request	*Else, if any check fails*
Loop to begining of this phase (ready for next anonymous authentication request)		

Fig 7